Undefined 0 (0) 1
IOS Press

Solving Guesstimation Problems Using the Semantic Web: Four Lessons from an Application

Alan Bundy [a,*], Gintautas Sasnauskas [a] Michael Chan [a]
[a] *School of Informatics, University of Edinburgh,*
A.Bundy@ed.ac.uk, gintautas.sasnauskas@gmail.com, mchan@ed.ac.uk

Abstract. We draw on our experience of implementing a semi-automated guesstimation application of the Semantic Web, GORT, to draw four lessons, which we claim are of general applicability. These are:

1. Inference can unleash the Semantic Web;
2. The Semantic Web does not constrain the inference mechanisms;
3. Curation must be dynamic; and
4. Own up to uncertainty.

Keywords: Guesstimation, Semantic Web, inference, dynamic curation, uncertainty

1. Introduction

As a result of the advocacy of the Semantic Web and Linked Data in, for instance, [3,4], new data is being added to The Web of Linked Data at an exponential rate. If we add to this data that that can be mined from semi-structured, web-based sources, then the amount of potential data is even larger. For instance, the KnowItAll Project [6] claims to have formed a 6 billion item ontology by extracting RDF triples and rules from English statements on the web with a 90% accuracy[1].

Existing applications of this wealth of data have only scratched the surface of the possibilities. The purpose of this paper is to emphasise some of the previously unexplored potential directions. This potential is illustrated by an unusual application to the solving of guesstimation problems: the GORT system. More details about GORT, including a discussion of its version history, can be found in §2.3. 2.4

2. Guesstimation Using the Semantic Web

Guesstimation is the task of finding an approximate answer to a quantitative problem based on a combination of intuition, facts, and reasoning [23,15]. Such problems are sometimes called *Fermi problems*, after Enrico Fermi who had been known to use the technique.

An example guesstimation problem is:

- How much would it cost to meet all the UK's electricity demand by solar panels?
- What area would they cover?

The number of solar panels required can be guesstimated by dividing the UK's electricity demand by the capacity of a typical solar panel. The cost can then be

*The research reported in this paper was supported by ONR project N000140910467 and EPSRC project EP/J001058/1. We would like to thank two SWJ referees: Simon Scheider and an anonymous referee.

[1] This is an estimate of the number of original English assertions that have been correctly formalised, not the accuracy of the original source information.

0000-0000/0-1900/$00.00 © 0 – IOS Press and the authors. All rights reserved

guesstimated by multiplying this number of panels by the cost of a typical panel. The area can be guesstimated by multiplying the number of panels by the area of a typical panel. More examples can be found in Table 1 in §2.3.3.

Traditionally, human guesstimators have to supply facts, such as the UK's electricity demand, and the capacity and area of a typical solar panel from their own background knowledge. We are now in the lucky position, however, that many of these facts can be retrieved from the Semantic Web using a search engine, such as SINDICE[19].

2.1. The Formalisation of Guesstimation

Guesstimation requires a new calculus for both representing the restricted kind numbers permitted in answers and to reasoning with these numbers to infer new approximate answers from old ones. In this section we define this calculus, which we call *the* SINGSIGDIG *Calculus* [2].

2.1.1. The Single Significant Digit Calculus

According to [23], the normal form for guesstimation answers is a number, in SI units, in single significant digit form, $d \times 10^i$, where d is a digit from $1, \ldots, 9$ and i is an integer. Where quantities are not originally in this normal form, non-SI units must be converted to SI and numeric values must be approximated to the form $d \times 10^i$.

Definition 1 (SINGSIGDIG Normal Form)

- *Let* $\mathbb{R}^\sim = \{d.10^i | d \in \{1, \ldots, 9\} \wedge i \in \mathbb{Z}\}$ *be the type of normal form numbers.*
- *Let* $nf^\sim : \mathbb{R} \mapsto \mathbb{R}^\sim$ *be the function that converts a real number into its nearest single significant digit normal form.*
- *Let* $\mathbb{U}$ *be the type of* SI *units.*
- *Let* $\mathbb{P}$ *be the product type* $\mathbb{R} \times \mathbb{U}$ *and* $\mathbb{P}^\sim$ *be the product type* $\mathbb{R}^\sim \times \mathbb{U}$*, i.e.,* $\mathbb{P}$ *and* $\mathbb{P}^\sim$ *are pairs of numbers and units. Note that* $\mathbb{P}^\sim$ *is a subtype of* $\mathbb{P}$*.*
- $\forall \langle r, u \rangle : \mathbb{P}. nf^\sim(\langle r, u \rangle) ::= \langle nf^\sim(r), u \rangle$.

1.2

For instance, $nf^\sim(3.142) = 3.10^0$. We will usually write $\langle r, u \rangle : \mathbb{P}^\sim$ as ra, e.g., $4.10^1 w$ instead of $\langle 4.10^1, w \rangle$, where w is the symbol for watts. For dimensionless quantities the unit will be omitted.

2.1

Definition 2 (SINGSIGDIG Formulae) *Formulae in the* SINGSIGDIG *Calculus are first-order expressions whose domain of discourse consists of pairs of type* $\mathbb{P}^\sim$*, plus everyday objects and sets of such objects.*

Any function f *(or predicate* p*) defined on* $\mathbb{P}$ *can be abstracted into a function* $f^\sim$ *(or predicate* $p^\sim$*) defined on* $\mathbb{P}^\sim$*.*

Suppose, without loss of generality, that all arguments of type $\mathbb{P}$ *of function* f *(predicate* p*) are initial, i.e., that:*

$$f : \mathbb{P}^m \times \tau_1 \times \ldots \times \tau_n \mapsto \mathbb{P}$$
$$p : \mathbb{P}^m \times \tau_1 \times \ldots \times \tau_n \mapsto \mathit{bool}$$

where τ_i *is the type of the* i^{th} *non-*$\mathbb{P}$ *argument of* f *and* $n \geq 0$*.*

For every such function f *(predicate* p*), we define a corresponding* $f^\sim : \mathbb{P}^{\sim m} \times \tau_1 \times \ldots \times \tau_n \mapsto \mathbb{P}^\sim$ *(*$p^\sim : \mathbb{P}^{\sim m} \times \tau_1 \times \ldots \times \tau_n \mapsto \mathit{bool}$*) as follows:*

$$\begin{aligned} & f^\sim(r_1^\sim, \ldots, r_m^\sim, t_1, \ldots, t_n) \\ & ::= nf^\sim(f(r_1^\sim, \ldots, r_m^\sim, t_1, \ldots, t_n)) \\ & p^\sim(r_1^\sim, \ldots, r_m^\sim, t_1, \ldots, t_n) \\ & ::= p(r_1^\sim, \ldots, r_m^\sim, t_1, \ldots, t_n) \end{aligned} \quad (1.2)$$

For instance,

$$\begin{aligned} 2.10^{11} w \div^\sim 3.10^1 w/m^2 &= nf^\sim(2.10^{11} w \div 3.10^1 w/m^2)) \\ &= nf^\sim(\frac{2}{3}.10^{10} m^2) \\ &= nf^\sim(\frac{2}{3}.10^{10}) m^2 \\ &= nf^\sim(7.10^9) m^2 \end{aligned} \quad (2.1)$$

In particular, we define the equality predicate $=^\sim$: $\mathbb{P}^{\sim 2} \mapsto \mathit{bool}$ by abstracting $=$. Where the context makes clear that an approximate function (predicate) is being used, we will usually drop the $\sim$ superscript.

These definitions ensure that $f^\sim$ ($p^\sim$) is uniquely defined on its first m numeric arguments. In order to ensure uniqueness for the next n *non-numeric* arguments, we need to make the following assumption.

Assumption 1 *Similarity Assumption:*

For all functions $f^\sim$ *(predicates* $p^\sim$*) and sets* S*, to ensure that* $f^\sim(\ldots, S, \ldots)$ *(*$p^\sim(\ldots, S, \ldots)$*) is uniquely defined, we assume that:*

$$\forall s_1, s_2 \in S.\ f^\sim(\ldots, s_1, \ldots) =^\sim f^\sim(\ldots, s_2, \ldots)$$
$$(\forall s_1, s_2 \in S.\ p^\sim(\ldots, s_1, \ldots) \iff p^\sim(\ldots, s_2, \ldots))$$

For instance, if $Solar$ is the set of all solar panels, then $power^{\sim}(Solar)$ is uniquely defined iff any pair of solar panels in $Solar$ have equal $power^{\sim}$s, i.e., their outputs in watts are equal up to one significant digit. If this is not the case then guesstimates based on their power may be inaccurate.

We will frequently want to describe a typical element of a set. To formalise this[2], we will designate ϵS to be a typical representative element of the set S. All we are allowed to know about ϵS is that $\epsilon S \in S$, not which specific element it is. ϵ is used in the methods *law of averages*, *energy*, *rate of change*, *arbitrary object* and *generalisation*.

We will use upper-case letters to represent sets and lower-case letters to represent objects. We will sometimes use polymorphic functions which apply to both objects and sets of those objects. If S is a set, the semantics of $f(S)$ is $f(\epsilon S)$. An exception to this semantic rule is the function $\|\ldots\| : Set(\tau) \mapsto \mathbb{P}^{\sim}$ where $\|S\|$ returns the approximate number of elements in the set S.

2.1.2. Guesstimation Methods

To solve guesstimation problems, GORT uses a set of proof methods which reason over the SINGSIGDIG Calculus. All the methods were developed by GORT developers by abstracting from the solutions to guesstimation problems provided in [23] and similar sources.

Each method is represented as a second-order, conditional, approximate, rewrite rule of the form:

$$Cond \implies LHS \rightsquigarrow RHS,$$

where $\rightsquigarrow$ is an oriented version of $=^{\sim}$. Such rewrite rules are applied to guesstimation goals as follows. If $Cond$ is true and LHS matches a sub-expression of the current goal with substitution σ, then replace this sub-expression with $RHS\sigma$ to form the next goal.

We divide the methods into primary and secondary. Primary methods are called by the user using the GUI described in §2.3.2. Secondary methods are applied automatically by GORT to solve the sub-goals created by the secondary methods. This section only describes the primary methods in detail. For more information on the secondary methods see [2].

[2]Our notation is borrowed from Hilbert's ϵ operator [9], which he introduced as an alternative to quantification and an aid to proving consistency. Technical knowledge of this operator is not needed in this paper.

The Count Method: The *count* method is applicable in cases where a guesstimation question requires the approximate number of elements $\|Small\|$ in a set $Small$, such that the sum of a function g over the elements in $Small$ is equal to $f(big)$. An example is: *How many solar panels would be required to meet all of the* UK*'s electricity demand?*. Here, $Small$ is the set of solar panels, big is the UK, f is electricity consumption and $g(s)$ is the power of a solar panel s, both measured in watts.

We can formalise the *count* method as:

Definition 3 (The *count* method)

$$g(Small) \neq \emptyset \wedge f(big) =^{\sim} \sum_{s \in Small} g(s) \implies$$

$$\|Small\| \rightsquigarrow \frac{f(big)}{g(Small)}.$$

The preconditions of the *count* method are checked by user interaction; the user instantiates f and g to functions for which s/he believes that the preconditions are true and that assumption 1 holds. Note the implicit universal quantification over the function variables f and g. This makes the *count* method, and three of the other primary methods described in this section, essentially second-order.

The Total Size Method: The *total size* method is applicable in cases where a guesstimation question requires the sum of some function f over a set S. An example is: *What is the total cost of a set of solar panels?*, e.g., the set required to meet all of the UK's electricity demand. Here, S is the set of solar panels and $f(s)$ is the cost of a solar panel s.

We can formalise the *total size* method as:

Definition 4 (The *total size* method)

$$\sum_{s \in S} f(s) \rightsquigarrow f(S) \times \|S\|,$$

where S is a set of non-numeric objects of type τ and f is a function $f : \tau \mapsto \mathbb{P}^{\sim}$.

The Law of Averages Method: The *law of averages*[3] method uses the fact that, on average, the proportion of times an object has a given property is equal to the proportion of objects in a larger population with that property at a given time. For example, the proportion

[3]This name is adopted from [23]. The normal pejorative use of this phrase is not intended.

of time that an average person spends asleep is equal to proportion of people on Earth asleep at any time, where S is the set of people, T is a finite set of equal time intervals in a day, and $\phi(s,t)$ asserts that person s is asleep during time interval t.

The *law of averages* method can be formalised as:

Definition 5 (The *law of averages* method)

$$S \neq \emptyset \wedge T \neq \emptyset \implies \frac{\|t \in T | \phi(\epsilon S, t)\|}{\|T\|} \rightsquigarrow \frac{\|s \in S | \phi(s, \epsilon T)\|}{\|S\|}.$$

Using this method, information about a wider population at a particular time can be found from an arbitrary, representative object over a period of time.

The Distance Method: The *distance* method is a domain-specific technique for calculating the distance between two locations on Earth. It applies in the case of a problem such as, *How much time would it take to drive from London to Manchester?*, where two locations are given and a distance is required.

The *distance* method can be formalised as:

Definition 6 (The *distance* method)

$$\Delta\widehat{\sigma} \rightsquigarrow 2\arcsin\left(\sqrt{\sin^2\left(\frac{\Delta\phi}{2}\right) + \cos\phi_s \cos\phi_f \sin^2\left(\frac{\Delta\lambda}{2}\right)}\right).$$

For two points $\langle \phi_s, \lambda_s \rangle$ and $\langle \phi_f, \lambda_f \rangle$, where the ϕs represent latitudes and λs represent longitudes, the planar angle between the points is calculated from the above formula as $\Delta\widehat{\sigma}$. Then the distance along the surface of the Earth is $r \cdot \Delta\widehat{\sigma}$, where the single significant digit approximation of r, the radius of the Earth, is 6×10^4 km.

Simon Scheider has suggested using navigation services to improve the accuracy of distance measures. This is an interesting suggestion, which we plan to in-
1.3 vestigate as further work.

The Energy Method: The *energy* method is used to calculate the total kinetic, potential, chemical or latent energy of a set of objects S.

To calculate the total energy, the method requires two parameters and a constant. Both parameters and the constant depend on the type of energy measured. The constant must be set to $c = 1$ when calculating latent or chemical energy, $c = \frac{1}{2}$ when calculating kinetic energy and $c = 10m/s^2$ when calculating potential energy. Furthermore, a fourth parameter, $\|S\|$, is required to be supplied which indicates the size of the set S. These four parameters are multiplied together.

For example, the following set of substitutions could be used to answer the question *How much potential energy does an average skyscraper have?*, where potential energy is calculated using the formula $E = mgh$.

- $c = 10m/s^2$, which is gravitational acceleration (g) on the Earth's surface;
- $f(\epsilon S)$, which is the mass of an typical skyscraper;
- $g(\epsilon S)$, which is the height of the centre of mass of a typical skyscraper, which can also be defined as half of the height of the skyscraper;
- $\|S\| = 1$, which stands for the number of skyscrapers we are interested in. If we were interested in finding the potential energy of *all* skyscrapers in Manhattan, then the number of skyscrapers in Manhattan would be used instead.

The *energy* method can be formalised as:

Definition 7 (The *energy* method)

$$E \rightsquigarrow c \times f(\epsilon S) \times g(\epsilon S) \times ||S||.$$

Rate of change method: The *rate of change* method is used to calculate the rate of change, when the size and the duration of the whole task is known. Alternatively it can be used for calculating the duration of the task when the size of the whole task and the rate of change is known. An example is *How long would it take to fill St Paul's Cathedral with water using a bucket?*. Here S would be a set of bucket loads of water needed to fill the Cathedral, $f(s)$ would be the time taken to pour one bucket load of water, s, into the Cathedral and $g(s)$ would be the volume of a bucket load of water, so $\sum_{s \in S} g(s)$ is the volume of the Cathedral.

The *rate of change* method can be formalised as:

Definition 8 (The *rate of change* method)

$$\sum_{s \in S} f(s) \rightsquigarrow \frac{f(\epsilon S) \times \sum_{s \in S} g(s)}{g(\epsilon S)}.$$

Get Data: The methods above break an initial question into sub-questions. Eventually, they reach questions that can be retrieved from the Semantic Web, e.g., using a semantic web search engine. This is done by the *get data* method. It constructs an RDF query consisting of a subject, predicate and a variable for the object. During data retrieval, the object is instantiated to the answer to the question. The latest version of GORT version 4.0, uses the SINDICE search engine. 2.4

The User Input Method: Unfortunately, the solution to a guesstimation question may require data that SINDICE *cannot* find on the Semantic Web. The *user input* method then asks the user for the required value.

Secondary Methods In addition to these primary methods, a set of secondary methods are applied automatically by GORT. These secondary methods are used to solve open subgoals arising from the primary methods. The secondary methods are just briefly mentioned below. More details can be found in [2,16].

The Arbitrary Object Method uses the ϵ operator to convert the value of some function of a set into the value of that function on a typical member of that set. This is formalised as:

$$f(S) \rightsquigarrow f(\epsilon S)$$

For example, S might be the set of humans and $f(s)$ the height of the human s.

The Average Value Method guesstimates a numeric value for some $f(\epsilon S)$ by computing the arithmetic mean of all $f(s), s \in S$: This is formalised as:

$$S \neq \emptyset \implies f(\epsilon S) \rightsquigarrow \frac{\sum_{s \in S} f(s)}{\|S\|}.$$

The Aggregation over Parts Method guesstimates the value of an attribute of a large object composed of many non-overlapping parts by summing the values for each of these parts. This is formalised as:

$$f(o) \rightsquigarrow \sum_{p \in Parts(o)} f(p),$$

where $Parts(o)$ is a function that returns the set of all non-overlapping parts of o.

The Generalisation Method guesstimates the average value of an attribute of the objects in a set by returning the average value of the attribute in a superset. This is formalised as:

$$S \subset T \implies f(\epsilon S) \rightsquigarrow f(\epsilon T)$$

The Geometry Methods calculate one attribute of a physical object from its other attributes. For instance, the circumference, $Circ(s)$, of a circular object, s, can be calculated from of its radius, $Radius(s)$ by:

$$Circ(s) \rightsquigarrow 2\pi Radius(s)$$

2.2. Worked Example

To illustrate the rewriting process used by these methods we apply them to the guesstimation problem:

How many solar panels would be required to meet all of the UK*'s electricity demand?*

The goal can be formulated as $\|Solar\|$, where $Solar$ is the set of solar panels required. As discussed in §2.1.2, in the preamble to Definition 3, the *count* method is applicable to problems of this sort. If we instantiate $Small$ to be $Solar$, big to be uk, f to be $eleccons$ and g to be $power$ then the *count* method is instantiated to:

$$power(Solar) \neq \emptyset \wedge$$
$$eleccons(uk) =^{\sim} \sum_{s \in Solar} power(s)$$
$$\implies \|Solar\| \rightsquigarrow \frac{eleccons(uk)}{power(Solar)}.$$

The user is currently required to confirm the correctness of the two conditions and assumption 1 in this case.

The *get data* method can now be used to find the values of both $eleccons(uk)$ and $power(Solar)$, which are returned as $2.10^{11}w$ and $4.10^{1}w$, respectively. The calculation can then be continued.

$$\begin{aligned}\|Solar\| &\rightsquigarrow \frac{eleccons(uk)}{power(Solar)} \\ &\rightsquigarrow \frac{2.10^{11}w}{4.10^{1}w} \\ &\rightsquigarrow 5.10^{9}\end{aligned}$$

to give the required answer, i.e., about 5 billion solar panels.

Questions 8 and 9 in Table 1 are variants of this question, both requiring a second application of the *count* method. Question 8 is also used as the example in the illustration of GORT's GUI in Figure 1. 2.1

2.3. The GORT System

GORT (*Guesstimation with Ontologies and Reasoning Techniques*) is a semi-automatic guesstimation system implemented in SWI-Prolog and Java. It has been developed in successive stages via four student projects.

GORT **1.0:** In his 2008-9 MSc project [1], Jonathan Abourbih built the initial version. It used lo-

cally stored RDF ontologies for looking up facts. Queries for the system had to be provided in the Prolog language, which limited the usability of the system.

GORT **2.0:** In his 2009-10 UG4 project [5], Luke Blaney added a web front-end, which allowed constructing queries by using a drag-and-drop interface. Furthermore, he replaced the majority of local data sources with remote RDF databases accessible via SPARQL end points. [2] describes GORT 2.0.

GORT **3.0:** In his 2010-11 MSc project [22], Yanyang Wang introduced new proof methods, which were able to solve a larger variety of problems. Furthermore, new compound units were constructed from atomic ones, e.g., m/s from m and s.

GORT **4.0:** In his 2011-12 UG4 project [16], Gintautas Sasnauskas improved the data retrieval by switching to the SINDICE search engine and adding a query decomposition module that broke queries into sub-queries. SINDICE retrieved more data, but this included more noisy data, so answer filtering was used to deals with multiple, possibly conflicting, values. Since units were often missing from RDF triples, he developed a method for guessing missing units. Subsequently, he has implemented a method for assigning an uncertainty value to the guesstimates.

2.3.1. The SINDICE *Semantic Web Search Engine*

The SINDICE Semantic Web search engine [19] was chosen for data retrieval in GORT 4.0 because is gives access to data from a large variety of sources. Furthermore, the SINDICE API presents RDFs grouped into documents or small ontologies covering single topics. This allows GORT to not only look for individual triples but also for documents containing information about an object using keyword search. An assumption was made that all RDF triples in a document which was found using keyword search describe the keyword. Following empirical observations, this assumption was found to be correct in the majority of cases.

2.3.2. The GUI

Users interact with GORT using a drag and drop, graphical user interface, which is illustrated in Figure 1. Users build up a query as a tree structure in which each node corresponds to a primary method, whose daughters provide its inputs, if any. The user builds this tree by dragging and dropping methods from the left-hand margin. In the case of the *get data* and *user input* methods, the user is prompted to supply additional information, namely subject plus predicate and value plus unit, respectively. As each method receives its inputs, it asynchronously updates itself and any methods that it is inputting too, allowing users to work on other parts of the tree in parallel. Completed methods turn from black to green and display their respective results. Results are presented in nodes as the preferred value, e.g., $2.10^{11}w$, together with:

- their error bounds, indicated as an interval, e.g., $[2.10^8, 3.10^{12}]$, and
- for *get data* method nodes, the number of data points in that interval, e.g., 10 and 7 in Figure 1, which we call the *rarity value*.

The *preferred value* of *get data* method nodes is either the mode, if a threshold is exceeded, otherwise the median (see §3.3.1 for more details). The preferred value of non-leaf nodes is calculated from the preferred value of their daughters. For more discussion of these values see §3.4. Sample queries are supplied on the right-hand side, to give the user exemplars of GORT's working. Commands are also supplied on the right-hand side for providing help, saving queries, resetting the system and closing it.

2.3.3. Evaluation of GORT

GORT 4.0 was evaluated on a test set of 12 examples [16]. Earlier versions of GORT were similarly evaluated on different test sets [1,5,22]. The list of problems is given in Table 1 and the results are summarised in Table 2

Since GORT 4.0 is an interactive system it is only meaningful to give timing data for sub-processes that are fully automated, such as queries to the SINDICE API. The subsequent calculations made with the retrieved data are just simple arithmetic and are neglible compared to the retrieval times. Typically, an initial query is made to SINDICE which takes an average of 2 seconds, and returns a document. This document is then cached, and typically about 8 further queries are made to it to retrieve particular parts of the document, which each take an average of 0.5 seconds. The complete procedure, therefore, typically takes about 6 seconds. Queries involving query decomposition usually take longer because they require a larger number of SINDICE queries. GORT 4.0 is much slower than GORT 3.0, because SINDICE is able to access far more sources of data than SPARQL. Some sample complete SINDICE query timings are summarised in Table 3.

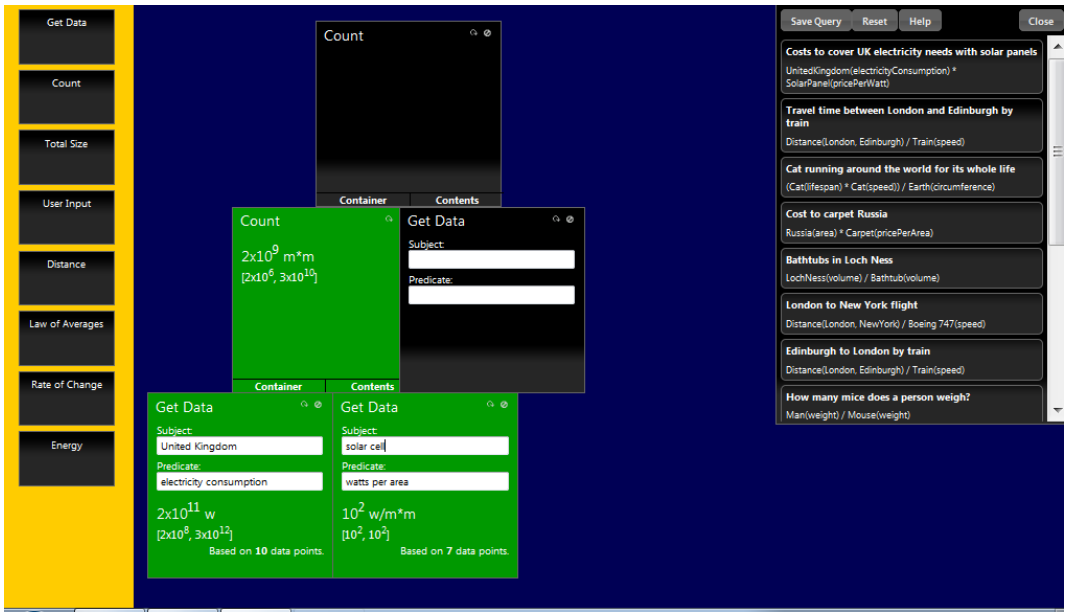

Fig. 1. **The** GORT GUI**:** The screenshot shows an intermediate stage in the solution of question 8 from Table 1. The left-hand margin contains buttons for each of the primary methods. The centre contains a tree representing the current query. Each node represents a proof method whose calculation, when complete, consists of the preferred value, the error interval and, for *get data* method nodes, the rarity value expressed as the number of data points found. The right-hand margin contains various commands and some example queries.

3. Lessons Learnt

In this section we draw some lessons from the experience of automating guesstimation. We argue that these are general lessons that apply to other applications of the Semantic Web, so should be taken into account by the developers of these applications.

3.1. Lesson 1: Inference can Unleash the Semantic Web

The Text Retrieval Conferences (TREC) (`http://trec.nist.gov/`) host a number of workshops, each focusing on a different information retrieval (IR) research area. At each conference competitions are run for information retrieval in each of these areas. The task in each competition is, given some search term, to find just those online documents that are relevant to the search term. Scores are based on recall and precision measures, i.e., the percentages of all and only the relevant documents found. Note that information is merely retrieved in the form that it was originally stored. There is no attempt to discover new information by old combining information from different sources.

It used to be the case that the field of question answering *did* infer new information from old. If the Wikipedia overview of this field (`http://en.wikipedia.org/wiki/Question_answering`) is accurate, however, then it too has degenerated into document and fact retrieval. Even the much celebrated IBM Watson system [18] focuses only on fact retrieval (see §3.5.1).

Information retrieval systems mostly search over online documents written in natural language. To enable inference on the information in these documents they need to be formalised, e.g., into RDF triples or description logic. There are several projects that are attempting to do this on a large scale, e.g., the KnowItAll Project [6], which claims to have automatically constructed an ontology of six billion triples, with 90% accuracy, from online natural language sources.

The Semantic Web consists of a huge number of formalised ontologies, using databases, taxonomies, RDF triples, description logics, etc. It thus has the potential to support inference to infer new information from old. Most Semantic Web search engines, however, such as SINDICE, are also restricted to fact retrieval[4]. Most in-

[4]Data-mining is an exception in that it does try to discover new information by looking for patterns in old information. The machine learning techniques used for data-mining are, however, outside the scope of the current discussion.

ID	Question
1	How many cars would we need for a bumper-to-bumper traffic jam from Edinburgh to Glasgow? (assuming one-lane highway)
2	How many baths of water are there in Loch Ness?
3	How many buffalo equivalent of meat does a typical dog eat in his lifetime?
4	How much fuel per person is used flying a Boeing 747 from London to New York?
5	How many cups of water would it take to fill a double decker bus?
6	How long would current uranium reserves last if all electricity was produced using nuclear power plants?
7	How much peak electricity could we generate if we covered the entire Sahara desert with solar panels?
8	What proportion of the UK would need to be covered by solar panels to cover all the UK's electricity needs?
9	How much would it cost to meet all the UK's electricity needs with solar panels?
10	What proportion of the UK would need to be covered by wind turbines to cover all the UK's electricity needs?
11	What is the total potential energy of rainfall which falls on the UK?
12	How many wind turbines would we need to cover the electricity needs of the UK?

Table 1
Questions used in evaluation of GORT 4.0.

Subject	Predicate	Time
Automobile	length	5.7 sec
Boeing 747	weight	2.9 sec
Cup	volume	11.7 sec
Solar panel	wattage per area	10.4 sec

Table 3
Some typical complete SINDICE query timings.

ference systems are restricted to reason over individual databases or ontologies, custom-built for a particular application.

Of course, it's very useful to be able to access any of the information that other people know and have stored online. How much more wonderful, though, to discover *information that no one knows*, by combining known information from diverse sources in new ways. GORT is one of several systems that show that this is possible. Some similar systems are discussed in §3.5.

Some of GORT's discoveries are novel only because no one would seriously be interested in knowing them, e.g., the number of cups of water needed to fill a double decker bus. But, as some of the questions in Table 1 illustrate, guesstimation can also be used, for instance, to give ball-park estimates of the environmental and financial impact of diverse solutions to the energy and climate crises. As MacKay has argued, [13], more widespread knowledge of these estimates are essentially to better inform the debate. Indeed, Mackay's book inspired us to tackle these guesstimation problem.

3.2. Lesson 2: The Semantic Web does not Constrain the Inference Mechanisms

RDF triples and the OWL description logic are both W3C standards. Most ontologies on the Semantic Web are formalised as databases or RDF and many of the rest in OWL. This has created a presumption that inference over the Semantic Web will be conducted using a logic based on SQL, RDF or in OWL. The GORT Project illustrates, however, that this need not be the case. GORT uses second-order, SINGSIGDIG Calculus proof methods §2.1.2 to reason about SINGSIGDIG formulae §2.1.1. These proof methods work by a second-order, rewriting process that would be very difficult, if not impossible, to formalise in a description logic. This is despite the use of SINDICE to collect the atomic facts in the form of RDF triples. What is going on?

GORT mines the Semantic Web for the information it needs, curates that information and then uses it dynamically to custom-build a local ontology tuned to the guesstimation problem it is currently trying to solve. It takes this opportunity to reformalise the RDF into SINGSIGDIG Calculus format, so that it can be reasoned with by SINGSIGDIG methods.

This solution is a general one. Developers of applications using the Semantic Web are free to use whatever inference engine best suits their application. Standard web search engines can be used to collect data in source formats. Dynamic curation §3.3 can then be used to create a custom-built, domain-specific ontology in the target format.

3.3. Lesson 3: Curation must be Dynamic

The information stored in the Semantic Web is noisy. There is no central control over who enters information into it and it would defeat its raison d'être to try to impose such control. Although there are W3C representation standards, information be represented in a wide variety of formats. Errors can occur due to carelessness, omission, ignorance, deliberate misinforma-

tion, etc. In addition, errors may be introduced during the retrieval process.

By way of illustration, in a recent mini-project to detect errors in the automatically constructed KnowItAll ontology [6], we discovered triples for the capital of Japan with the following answers: Tokyo, Kyoto and Paris [8]. "Tokyo" is the correct answer; "Paris" came from a tutor on logic and exemplified a false assertion; "Kyoto" used to be the capital of Japan but was curated to the present tense as a side effect of retrieval. In this case, "Tokyo" was by far the most popular answer retrieved, so could be selected by taking the mode, giving a reliable answer from noisy data. This was not always the case. Therefore, before it can be used, information must be curated to remove errors, put it in a common format, etc. By *static curation* I mean the pre-processing of the whole or a large part of the information stored in the Web to construct an ontology that can then be used for retrieval, such as proposed in [6]. Static curation is unrealistic for several reasons.

- The Semantic Web is huge: in 2011, the Web of Linked Data was estimated at 32 billion RDF triples[5]. Curating it all would be a mammoth under-taking and impossible without significant automated assistance.
- It is growing very fast: in 2011 The Web of Linked Data was estimated was estimated to have doubled every year since it's creation [6]. It will be impossible to keep up with this exponential growth. A (semi-)automated solution that just works in one year will fail in the next.
- Curation is application-specific. There is no one curation fits all. In §3.3.2, we give an example of this arising from our project.

The answer is to curate *dynamically*, i.e., selectively and on an as-needed basis for the current application in an application-specific way. In §3.3.1, we outline how this was done in the GORT Project.

3.3.1. GORT*'s Dynamic Curation*

GORT forms a custom-built, locally-stored ontology for each question. Forming this local ontology involves the following process, which includes different kinds of dynamic curation.

- The ontology is initialised with GORT's general-purpose proof methods, such as those outlined in §2.1.2.
- The guesstimation problem is formalised as a goal to be proved and these proof methods are applied to it in a top-down process to grow the proof as a tree, as illustrated in Figure 1.
- The leaves of this tree are factual queries to be solved by the *get data* or *user input* methods. In the case of *get data* applications, SINDICE is used to retrieve the relevant facts and these are subject to curation with the aim of augmenting the local ontology. Facts input via *user input* are also curated and added to the ontology.
- Curation includes:
  - normalisation into the SINGSIGDIG Calculus;
  - abstraction of some concrete subjects into generic ones; and
  - augmentation with additional information.
  - where, even after normalisation, SINDICE provides multiple different values, two forms of averaging are used to construct one preferred value.

  These forms of curation are further discussed in §3.3.2

When multiple values are returned by SINDICE GORT must use them to construct a single preferred value. Firstly, normalisation to the SINGSIGDIG ensures that minor differences between values disappear. If there is still a list of multiple values (duplicates included) after normalisation then two kinds of averaging are attempted in turn: mode and median.

Mode: The mode is the number that appears most often in a list of numbers. If one value is returned at least twice more often than the second most common value, then GORT assumes that this is the correct answer and that the other values are erroneous and can be ignored.

Median: Otherwise the preferred value is the median of the ordered list of returned values. The median is the value that breaks this ordered list into two equal sized sub-lists. If the list has an odd number of values then it will be the middle value; it has an even number then it will be the arithmetic mean of the two middle values. The median is preferred over the arithmetic or geometric mean because it undervalues outliers. One large outlier can disproportionately skew either mean. Since outliers are

[5] http://events.linkeddata.org/ldow2012/

[6] http://www.readwriteweb.com/cloud/2011/01/the-concept-of-linked-data.php

Subject	Predicate	Value
dbr:Dino_automobile	dbpp:length	166.25
dbr:Astra_(1954_automobile)	dbpp:length	114
dbr:Mercedes-Benz_W201	dbpp:length	4420
dbr:Ford_Taurus_(third_generation)	dbpediaowl:length	5.0165
dbr:Xtra_(automobile)	dbpp:length	106

The top 5 results from SINDICE *context-sensitive search for triples in format* $(*, length, X)$ *in documents found using the keyword "automobile".*

Table 4
Lengths of automobiles

quite likely to be erroneous, then it is better to use the median.

3.3.2. *Application-Specific Curation*

As an example of application-specific curation from the GORT Project, consider Table 4.

This table shows some of the information retrieved by SINDICE to answer the guesstimation question "How many cars would we need for a bumper-to-bumper traffic jam from Edinburgh to Glasgow?". To answer this question it was necessary to guesstimate the length of a typical car. This kind of generic information is rarely stored directly in the Semantic Web. What is stored, however, is information on the length of particular makes and models of cars: indeed the manufacturers are keen to supply this information on their web sites. To obtain this model-specific information GORT takes advantage of the context-sensitive search feature of SINDICE. Firstly, GORT chooses an appropriate context: *automobile* in this case. Then GORT constructs the wild-card triple $(*, length, X)$ and asks SINDICE to use this as a query in the automobile context.

So, the first step of application-specific curation necessary for GORT to make use of this data is *abstraction*: all the different car makes and models in the subject heading of Table 4 must be abstracted to 'automobile'. This abstraction is application specific because, for a different problem, it might be important to *preserve* the details about make and model. Indeed, it might be necessary to *refine* it, e.g., by distinguishing the different the years of manufacture of each model.

Note that the various values in Table 4 are numbers without units. From the diversity of these numbers it is clear that these lengths are expressed in different units, e.g., centimetres, feet, etc. The unit used is implicit in the source ontology of the triple, e.g., a particular manufacturer might always express distances in centimetres. The various numbers cannot be compared unless they are translated into the same units, e.g., SI units, which in this case is metres.

This kind of omission is commonplace in RDF triples. The triple format limits the amount of information that can be recorded. This limitation could be overcome if compound entries were normally permitted, e.g., a pair of number and unit, but compound entries are not normally permitted. Alternatively, there is a standard translation of an n-ary relation into $n + 1$ triples: $rel(t_1, \ldots, t_n)$ is represented as $(rel_0, arg_1, t_1), \ldots, (rel_0, arg_n, t_n)$, where rel_0 names this particular n-ary relation and $arg - i$ is the name of the i^{th} argument of rel. This is the method used by RDF Schema, but it has not been widely adopted. Ontologies specifically for describing units also exist, for instance, [20], but they also are not widely known or used. Users of the Semantic Web are not in a position to dictate or enforce such standards. We can use only what is provided and cope with its inherent vagueness, incompleteness and uncertainty. 1.4

To overcome the omission of units, we have employed a heuristic based on the ratios of the numbers, especially between imperial and metric units. Note that the ratios between these units is often unique, e.g., that between centimetres and feet. If we assume (as we do, see Assumption 1) that the normalised values of the length of the different car makes are equal, then we can often work out what the units must be. Consider,

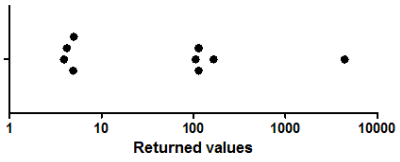

Fig. 2. **Clustered values for the length of a car**

Fig 2. The x-axis[7] is a logarithmic scale representing the car lengths returned by SINDICE, some of which are given in Table 4. GORT scores a variety of models against such clusters. For the values in Fig 2 the best fitting model is millimetres, inches and metres. Converting all the numeric values according to this model and then taking the median gives a typical car length in SI units of 3.909 metres. The final stage of curation is to normalise this value to 4.10^1 metres.

[7]The y-axis is only used to separate the dots and has no significance.

3.4. Lesson 4: Own up to Uncertainty

As discussed in §3.3, information retrieved from the Semantic Web is inherently noisy.

It is important that users of information retrieved from the Semantic Web know how much trust to put in the answers. There is an obligation, therefore, to own up to the uncertainty in an answer and to try to estimate how uncertain it is. In GORT we decided to indicate uncertainty by providing the range and diversity of values that SINDICE obtained for each fact and the consequences as these facts were propagated by inference. The range and diversity were summarised in two ways:

- An interval is used to indicate the error bounds, i.e., the minimum and maximum values, e.g., $[2.10^8, 3.10^{12}]$.
- A rarity value is used to indicate how many value instances were found.

Ideally, the same (hopefully correct) value will be found many times and no other values (presumed erroneous) will be found. This ideal situation will be indicated by a high rarity value and an error bound interval covering only one value, i.e., $[n, n]$. The wider the interval and the lower the rarity value the more uncertain the preferred value is to be correct.

This is only a heuristic. For instance, when SINDICE was used find the diameter of a golf ball, we got lots of values of 6.10^{-2}, even through the correct answer is 4.10^{-2}. This is because SINDICE found lots of references to the diameter of golf ball light bulbs and none to real golf balls. Never-the-less, the heuristic worked well in most cases.

Other ways of indicating uncertainty are, of course, possible. For instance, we also considered trying to estimate a probability of correctness. We also thought of giving a distribution, e.g., a gaussian, of the values instead of just an interval in order to indicate which values were most common. In the end we decided that the interval and rarity value provided the best balance between explanatory content and understandability. Of course, this solution is geared to the return of numerical values. For more qualitative values, the interval would not work, although the rarity value would. We only advocate that some uncertainty indication be provided as a basis for user trust.

3.5. Related Work

In this section we compare GORT with related systems on the dimensions of:

SW: Whether they retrieve information in a closed way by accessing only fixed, pre-stored knowledge bases (✗), or in an open way from the Semantic Web or similar open knowledge source (✔).

Inf: Whether they just retrieve information that is already represented in the knowledge source (✗), or are able to infer new knowledge not already represented (✔).

Uncon: Whether they are constrained to the inference mechanisms provided for the formalism in which their knowledge source is represented (✗), or whether they employ additional inference mechanisms, perhaps by translating the retrieved knowledge to a different formalism (✔).

DynCur: Whether the knowledge not curated or is curated statically (off-line) (✗), or whether it is curated dynamically, as required for the current application (✔).

Uncert: Whether some measure of the uncertainty is provided for the knowledge retrieved or inferred (✔), or not (✗).

These dimensions are based directly on the four lessons which form the claims of this paper. Following the discussion of each related system, the results are summarised in Table 5. Unfortunately, it is not possible to give a quantitative comparison, as no other system solves the same problems as GORT. 1.1

Given the huge number and variety of related systems we have restricted this discussion to some of the foremost representatives of the principle classes of these systems.

IBM**'s Watson:** was chosen to represent the class of information-retrieval systems.

BotE-Solver: was chosen to represent the class of guesstimation systems.

QUARK: was chosen to represent the class of question-answering systems.

We believe that the conclusions reached for each representative system also broadly hold for the class of systems that it is representing.

A comparison of some of the same related systems on four different dimensions can be found in [2], which describes GORT 2.0. This previous related work comparison includes QUARK and the following four other systems:

Power-Aqua a Semantic Web-based question answering system [12];

CS Aktive Space a system for tracking UK computer science research [17];

Cyc a general-purpose 'common-sense' reasoning system [11]; and

Wolfram|Alpha a system that calculates answers to numerical questions on a wide range of topics[8].

3.5.1. IBM's Watson

Watson was originally built to compete in the US Jeopardy!™ competition, where three contestants try to answer general knowledge questions and puzzles [7]. It famously beat the two best human players in a nationally televised Jeopardy! competition on 14^{th} January 2011. The underlying technology of Watson is now being applied to more practical challenges, e.g., in health care.

Watson employs a very wide range of complementary natural language processing and information retrieval techniques orchestrated by a massively-parallel architecture, called DeepQA, and running on a large bank of high-performance servers. It generates many candidate answers for each question and assigns each a score, which estimates the probability that this answer is correct. It buzzes in and provides the highest scoring answer only if it exceeds a dynamically calculated threshold. The rules of Jeopardy! preclude it from consulting external information sources, so its knowledge is pre-stirred using a wide range of knowledge bases, i.e., it does not access any open knowledge source, such as the Semantic Web, although it could probably be readily adapted to do so. Watson does not currently reveal its degree of uncertainty in the answer, but it could be easily adapted to do so, if this was appropriate for the application.

Only about 2% of the required answers to questions are only found directly in a knowledge source. So, Watson uses various forms of curation and inference to combine information from different knowledge sources into the required answers. These inference mechanisms include type coercion, geospatial and temporal constraints, rule-based reasoning, and statistical and machine learning techniques, as well as specialised techniques for particular kinds of puzzles, puns and missing links [10].

3.5.2. Back of the Envelope Reasoning

The BotE-Solver solves guesstimation problems using seven strategies [14]. These strategies are based on:

Analogical reasoning: to transform solutions to previous problems into solutions to the current one, and

Qualitative reasoning: to abstract exact values into order-of-magnitude approximations.

It uses the CYC knowledge base as a source of factual information [11]. It has been successfully tested on a corpus of 13 problems. Some of the BotE-Solver's strategies are similar to GORT's, e.g., its Mereology strategy is similar to GORT's *total size* in dividing an object into parts and summing the attributes of each part. Other strategies are very different, e.g., its Similarity strategy transforms an object into a similar one, say Australia into Switzerland as a way of estimating population size.

The BotE-Solver is similar to GORT in inferring new information from old and in being unconstrained in its inference mechanisms. Since it uses a statically curated knowledge base, it does not need to curate dynamically, as GORT does. It does not take advantage of the Semantic Web in retrieving information. Nor does it assign an uncertainty measure to its results. Instead, users are recommended to solve each problem in multiple ways in order to estimate the robustness of the results.

3.5.3. The QUARK *Deductive Question Answerer*

QUARK answers questions posed in English by combining knowledge from multiple sources [21]. Although it embodies several novel features, for the purposes of this comparison with related work, it can be regarded as typical of question-answering systems. It uses the GEMINI English parser to translate the input questions into a logical form and then the SNARK theorem prover to derive the required answer from pre-stored knowledge. It draws that knowledge from a variety of third-party ontologies, which include: the Alexandria Digital Library Gazetteer, the CIA Factbook and various NASA sources. It also uses the ASCS Semantic Web search engine, but that appears only to be used to search the above ontologies and not to do unbounded searches on the Semantic Web. Curation is used to translate between the formalisms of the various source ontologies and the internal formalism used by SNARK. The forms of inference used by SNARK are resolution, paramodulation and *procedural attachment*, which calls domain-specific procedures, e.g., for numeric calculation, on certain goal types. Procedural attachment is also used by SNARK to access the various third-party ontologies.

[8] http://www.wolframalpha.com/

System	SW	Inf	Uncon	DynCur	Uncert
Watson	✗	✔	✔	✔	✔
BotE-Solver	✗	✔	✔	✗	✗
QUARK	✗	✔	✔	✗	✗
GORT	✔	✔	✔	✔	✔

Table 5
Related work summary:

QUARK is similar to GORT in inferring new information from old and in being unconstrained in its inference mechanisms. Since it uses a statically curated ontologies, it does not need to curate dynamically, as GORT does. It does not take advantage of the Semantic Web in retrieving information. Nor does it assign an uncertainty measure to its results.

4. Conclusion

In this paper we have argued that to make best use of the Semantic Web we must accommodate the following four lessons:

1. Factoid retrieval only scratches the surface of what is possible. We want not just to retrieve *known* information from the internet, but combine it in novels ways to infer previously *unknown* information.
2. We can draw on a wide range of different inference mechanisms to infer this new information. We need not be constrained to those inference mechanisms associated with the format in which the retrieved information is stored, e.g., description logic decisions procedures. The retrieved information can, instead, be curated into whatever format is required by our preferred inference mechanism.
3. The curation of information must be dynamic, i.e., done at retrieval time in an application-specific way. This is not only because the Semantic Web is too big and growing too fast for static curation to be feasible. It is also because curation has to be application and inference mechanism specific.
4. The Semantic Web is very noisy. A large amount of its content is inaccurate or downright false. Uncertain results should not be presented to users as if their truth was guaranteed. Some measure of their uncertainty must be calculated and made available to the user, so they can judge how much faith to put into any decisions based on them. This may involve trying to answer a question in independent ways and comparing the different answers.

We have illustrated these four lessons by describing the GORT system, which we claim embodies all four of them. We have contrasted it with rival approaches. Only IBM's Watson provides a comparable illustration.

References

[1] Abourbih, Jonathan A. (2009). Method and system for semi-automatic guesstimation. Msc project, University of Edinburgh, Scotland.

[2] Abourbih, J.A., Blaney, L., Bundy, A. and McNeill, F. (2010). A single-significant-digit calculus for semi-automated guesstimation. In Giesl, Jürgen and Hähnle, Reiner, (eds.), *Automated Reasoning*, volume 6173 of *Lecture Notes in Computer Science*, pages 354–368. Springer.

[3] Berners-Lee, Tim, Hendler, James and Lassila, Ora. (May 2001). The Semantic Web. *Scientific American*, 284(5):34–43.

[4] Bizer, Christian, Heath, Tom and Berners-Lee, Tim. (2009). Linked data—the story so far. *International Journal on Semantic Web and Information Systems*. In press.

[5] Blaney, L. (2010). Semi-automatic guesstimation. Undergraduate project, University of Edinburgh, Scotland.

[6] Etzioni, O., Fader, A., Christensen, J., Soderland, S. and Mausam, M. (2011). Open information extraction: The second generation. In *Proceedings of IJCAI*, pages 3–10.

[7] Ferrucci, D. A. (May/July 2012). Introduction to "this is watson". *IBM J. Res. & Dev.*, 56(3/4):1:1–1:15.

[8] Gkaniatsou, A., Bundy, A. and McNeill, F. (2012). Towards the automatic detection and correction of errors in automatically constructed ontologies. In *8th International Conference on Signal Image Technology and Internet Based Systems*, pages 860–867.

[9] Hilbert, David and Bernays, Paul. (1939). *Die Grundlagen der Mathematik — Zweiter Band*. Number L in Die Grundlehren der Mathematischen Wissenschaften in Einzeldarstellungen. Springer.

[10] Kalyanpur, A. et al. (May/July 2012). Structured data and inference in deepqa. *IBM J. Res. & Dev.*, 56(3/4).

[11] Lenat, D.B. (November 1995). CYC: a large-scale investment in knowledge infrastructure. *Commun. ACM*, 38(11):33–38.

[12] Lopez, Vanessa, Guidi, Davide, Motta, Enrico, Peroni, Silvio, d'Aquin, Mathieu and Gridinoc, Laurian. (October 2008). Evaluation of semantic web applications. OpenKnowledge Deliverable D8.5, Knowledge Media Institute, The Open University, Milton Keynes, England, Accessed 10 August 2009.

[13] MacKay, D. J.C. (2009). *Sustainable Energy - without the hot air*. UIT Cambridge Ltd., Cambridge, United Kingdom.

[14] Paritosh, P.K. and Forbus, K.D. (2005). Analysis of strategic knowledge in back of the envelope reasoning. In *Proceedings of the 20th National Conference on Artificial Intelligence (AAAI-05)*.

[15] Santos, A. (2009). *How Many Licks: Or, How to Estimate Damn Near Anything*. Perseus Books.

[16] Sasnauskas, G. (2012). Semi-automatic guesstimation iv. Undergraduate project, University of Edinburgh, Scotland.

[17] Shadbolt, N., Gibbins, N., Glaser, H., Harris, S. and Schraefel, M.C. (2004). CS AKTive space, or how we learned to stop worrying and love the semantic web. *IEEE Intelligent Systems*, 19(3):41–47.

[18] Systems, IBM and Group, Technology. (February 2011). Watson — a system designed for answers: The future of workload optimized systems design. An IBM white paper, IBM.

[19] Tummarello, G., Delbru, R. and Oren, E. (2007). Sindice.com: Weaving the open linked data. In Aberer, K. et al, (ed.), *The Semantic Web*, volume 4825 of *Lecture Notes in Computer Science*, pages 552–565. Springer Berlin / Heidelberg.

[20] van Assem, M., Rijgersberg, H. and Top, J. (2013). Ontology of units of measure and related concepts. *Semantic Web*, 4(1):3–13.

[21] Waldinger, Richard J., Appelt, Douglas E., Dungan, Jennifer L., Fry, John, Hobbs, Jerry R., Israel, David J., Jarvis, Peter, Martin, David L., Riehemann, Susanne, Stickel, Mark E. and Tyson, Mabry. (2004). Deductive question answering from multiple resources. In *New Directions in Question Answering'04*, pages 253–262.

[22] Wang, Y. (2011). Semi-automatic guesstimation 3. Msc project, University of Edinburgh, Scotland.

[23] Weinstein, L. and Adam, J.A. (2008). *Guesstimation: solving the world's problems on the back of a cocktail napkin*. Princeton University Press.

ID	Subject	Predicate	Correct Answer	GORT 4.0 Answer	Range from	Range to	Comments
1	Automobile	Length	5.10^0 m	4.10^0 m	4.10^0	4.10^0	Values are available in variety of units, none explicitly defined.
2a	Bathtub	Volume	2.10^{-1} m^3	2.10^{-1} m^3	1.10^{-1}	2.10^{-1}	Values are available in gallons and litres, none explicitly defined; noise.
2b	Loch Ness	Volume	7.10^9 m^3	7.10^9 m^9	2.10^{-3}	7.10^9	Noise due to volume of different Loch Ness related products.
3a	Buffalo	Weight	1.10^3 kg	2.10^3 kg	1.10^{-1}	2.10^3	Weight of the heaviest recorded buffalo was retrieved.
3b	Dog	Lifespan	4.10^8 s	4.10^8 s	4.10^8	5.10^8	Average lifespans of different breeds of dogs are available.
4	Boeing 747	Fuel capacity per range	1.10^{-2} kg/m	5.10^{-3} kg/m	3.10^{-3}	5.10^{-3}	Value directly unavailable, must decompose into fuel capacity and range.
5a	Cup	Volume	2.10^{-4} m^3	2.10^{-4} m^3	1.10^{-4}	2.10^{-3}	Volume directly unavailable; must decompose into height and diameter.
5b	Double decker bus	Volume	9.10^2 m^3	2.10^{-3} m^3	2.10^{-3}	2.10^{-3}	Volume is directly unavailable; must decompose into width, height and length. Volume of toy double decker bus is retrieved.
6	Nuclear reactor	Capacity	3.10^9 W	3.10^9 W	1.10^9	5.10^9	Different values available for different nuclear reactors.
7a	Sahara Desert	Area	9.10^{12} m^2	2.10^{11} m^2	3.10^{11}	2.10^{12}	Area of Western Sahara is retrieved.
7b, 8c	Solar panel	Watts per area	1.10^2 W/m^2	1.10^2 W/m^2	8.10^1	1.10^2	Value directly unavailable; must decompose into watts and area. Area must be further decomposed into width and height.
8a, 10a, 11a	United Kingdom	Area	2.10^{11} m^2	2.10^{11} m^2	2.10^{11}	2.10^{11}	Accurate guesstimate.
9b	Solar panel	Price per watt	3.10^0USD/W	1.10^1USD/W	2.10^0	2.10^1	Value directly unavailable, must decompose into price and wattage.
11b	United Kingdom	Precipitation	9.10^{-1} m	4.10^0 m	4.10^0	5.10^0	Only value for the wettest place in the United Kingdom (Crib Goch) is available.e
12b	Wind turbine	Capacity	4.10^6 W	4.10^6 W	4.10^6	1.10^8	Well formatted values available for different wind turbines and wind turbine parks, moderate noise.

Table 2

Evaluation results for GORT 4.0. Correct answers were derived manually from a variety of publicly available sources.